

Grupo4 - FFT

De Física Computacional

A Transformada rápida de Fourier (em inglês **Fast Fourier Transform**, ou **FFT**) é um algoritmo que torna viável o cálculo da Transformada Discreta de Fourier (DFT), que é a forma discretizada da [Transformada de Fourier (https://pt.wikipedia.org/wiki/Transformada_de_Fourier)]. A **FFT** permite transformar de forma rápida uma série de sinais discretos em uma amostra contendo as frequências desses sinais, desde que satisfaça algumas propriedades.

*O sinal é
o fresh?*

Índice

- 1 Transformada Discreta de Fourier
- 2 Transformada Rápida de Fourier
 - 2.1 Exemplo
 - 2.2 FFT para N diferente de uma potência de 2
 - 2.3 Algorítmo (usando recursão)
- 3 Implementação
- 4 Exemplos
 - 4.1 Gaussiana
 - 4.2 Oscilador Harmônico Amortecido
- 5 Aplicações
 - 5.1 Equação de Schrödinger: propagação de onda de probabilidade
 - 5.1.1 Potencial Quadrado
 - 5.1.2 Potencial Triangular
 - 5.1.3 Potencial Batman
 - 5.2 Padrões de Difração
- 6 Referências Bibliográficas

Transformada Discreta de Fourier

Em muitas aplicações se tem informação sobre um conjunto de dados, ao invés de uma função contínua. A **Transformada Discreta de Fourier** transforma esse conjunto de dados em um conjunto de tamanho igual com informação sobre as frequências da função que satisfaz o conjunto de dados.

Para um conjunto de dados igualmente espaçados, pode-se, ao considerar os dados como um período de uma função periódica, cujo período normalmente é considerado entre $[-\pi, \pi]$ para facilitar o cálculo (e que pode sempre ser transformada em uma função nesse intervalo), mostrar que a transformada discreta de Fourier pode ser dada pela equação:

$$F_k = \sum_{n=0}^{N-1} f_n e^{-i2\pi nk/N}$$

*← ↗ Fator
1
2π?*

A sua inversa é, em paralelo ao caso da transformada contínua,

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} F_k e^{i2\pi nk/N}$$

A transformada também pode ser expressa em forma vetorial, como

$$\vec{A} = \mathbf{W}^{\text{nk}} \vec{a} \text{ onde } \mathbf{W}^{\text{nk}}$$

é base ~ de contiz...?

$$\mathbf{W}^{nk} = \begin{bmatrix} e^{-2\pi i 0 \cdot 0/N} & e^{-2\pi i 0 \cdot 1/N} & \dots & e^{-2\pi i 0 \cdot k/N} \\ e^{-2\pi i 1 \cdot 0/N} & e^{-2\pi i 1 \cdot 1/N} & \dots & e^{-2\pi i 1 \cdot k/N} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-2\pi i n \cdot 0/N} & e^{-2\pi i n \cdot 1/N} & \dots & e^{-2\pi i n \cdot k/N} \end{bmatrix}$$

O cálculo dessa expressão leva em torno de N^2 passos para o resultado. Uma amostra com 3,000 pontos precisa de 9,000,000 operações para a transformada ser obtida, tornando a DFT inviável para aplicações rápidas.

Transformada Rápida de Fourier

É possível calcular a transformada com $N \log_2 N$ passos. Para isso se dispõe de um algoritmo chamado **Transformada Rápida de Fourier**. Considera-se um conjunto de pontos $N = 2^p$ (com p inteiro, então, da definição da DFT)

$$F_k = \sum_{n=0}^{N-1} f_n \cdot e^{-i \frac{2\pi}{N} \cdot k \cdot n}$$

in fechado...

podemos dividir o somatório em 2:

$$F_k = \sum_{n=0}^{N/2-1} f_{2n} \cdot e^{-i \frac{2\pi}{N} \cdot k \cdot 2n} + \sum_{n=0}^{N/2-1} f_{2n+1} \cdot e^{-i \frac{2\pi}{N} \cdot k \cdot (2n+1)}$$

$$F_k = \sum_{n=0}^{N/2-1} f_{2n} \cdot e^{-i \frac{2\pi}{N/2} \cdot k \cdot n} + e^{-i \frac{2\pi}{N} k} \sum_{n=0}^{N/2-1} f_{2n+1} \cdot e^{-i \frac{2\pi}{N/2} \cdot k \cdot n}$$

$$F_k = \sum_{n=0}^{N/2-1} f_{2n} \cdot e^{-i \frac{2\pi}{N/2} \cdot k \cdot n} + C(k) \sum_{n=0}^{N/2-1} f_{2n+1} \cdot e^{-i \frac{2\pi}{N/2} \cdot k \cdot n}$$

onde a soma em vermelho é a parte **par** e a soma em azul é a parte **ímpar** da transformada. As duas somas tem o mesmo expoente, que agora é dividido por $N/2$. Desse expoente, é evidente a relação entre o ponto k e o ponto $k + N/2$

$$e^{-i \frac{2\pi}{N/2} \cdot k \cdot n} = e^{-i \frac{2\pi}{N/2} \cdot (k+N/2) \cdot n} = e^{-i \frac{2\pi}{N/2} \cdot k \cdot n} \cdot e^{-i \frac{2\pi}{N/2} \cdot N/2 \cdot n} = e^{-i \frac{2\pi}{N/2} \cdot k \cdot n} \cdot 1$$

Com essa relação, podemos ver que F_k e $F_{k+N/2}$ tem o mesmo expoente e podem ser calculadas ao mesmo tempo. Mais que isso, a nova forma da transformada pode ser sucessivamente dividida, cada vez produzindo somas com limites menores.

Exemplo

Suponha que temos a função sinusoidal $a(t) = \sin(2\pi \cdot 1Hz \cdot t)$ e fazemos quatro medidas no intervalo de 1 segundo, resultando em

$$a_0 = 0.00 \quad a_1 = 1.00 \quad a_2 = 0.00 \quad a_3 = -1.00 \quad \text{Separar...}$$

Com essas 4 medidas, podemos dividir a soma 2 vezes:

$$A_k = \sum_{t=0}^3 a_t \cdot e^{-i \frac{2\pi}{4} \cdot k \cdot t}$$

$$A_k = \sum_{t_1=0}^1 a_{2t_1} \cdot e^{-i \frac{2\pi}{2} \cdot k \cdot t_1} + C_k^1 \sum_{t_1=0}^1 a_{2t_1+1} \cdot e^{-i \frac{2\pi}{2} \cdot k \cdot t_1}$$

$$A_k = \sum_{t_2=0}^0 a_{4t_2} \cdot e^{-i\frac{2\pi}{1} \cdot k \cdot t_2} + C_k^2 \sum_{t_2=0}^0 a_{4t_2+2} \cdot e^{-i\frac{2\pi}{1} \cdot k \cdot t_2} + C_k^1 \sum_{t_2=0}^0 a_{4t_2+1} \cdot e^{-i\frac{2\pi}{1} \cdot k \cdot t_2} + C_k^3 \sum_{t_2=0}^0 a_{4t_2+3} \cdot e^{-i\frac{2\pi}{1} \cdot k \cdot t_2}$$

e como temos $C_k^j = (e^{-i\frac{2\pi}{N}k})^j$ podemos calcular

$$A_0 = 1.00 \cdot C_0^1 - 1.00 \cdot C_0^3 = 0.00 + i0.00$$

$$A_1 = 1.00 \cdot C_1^1 - 1.00 \cdot C_1^3 = 0.00 - i2.00$$

$$A_2 = 1.00 \cdot C_2^1 - 1.00 \cdot C_2^3 = 0.00 + i0.00$$

$$A_3 = 1.00 \cdot C_3^1 - 1.00 \cdot C_3^3 = 0.00 + i2.00$$

FFT para N diferente de uma potência de 2

Mesmo com a FFT sendo um algoritmo extremamente eficiente para $N = 2^p$, esse dificilmente é o caso que encontramos. Ainda assim, para N altamente composto ($N = r_1 \cdot r_2 \cdot \dots \cdot r_m$) o algoritmo ainda resulta em uma boa queda no tempo de cálculo.

Para o caso mais simples $N = r_1 \cdot r_2$ a transformada pode ser escrita como

$$F(k_1, k_0) = \sum_{n_0=0}^{r_2-1} \left[\sum_{n_1=0}^{r_1-1} x(n_1, n_0) e^{-i2\pi \cdot k \cdot n_1 \cdot r_2} \right] e^{-i2\pi \cdot k \cdot n_0}$$

onde

$$k = k_1 \cdot r_1 + k_0 \quad k_0 = 0, 1, \dots, r_1 - 1 \quad k_1 = 0, 1, \dots, r_2 - 1$$

$$n = n_1 \cdot r_2 + n_0 \quad n_0 = 0, 1, \dots, r_2 - 1 \quad n_1 = 0, 1, \dots, r_1 - 1$$

assim a transformada que antes necessitava de N calculos, agora pode ser vista como $r_1 + r_2$ calculos.

Algoritmo (usando recursão)

```
>> function FFT(N, d)
>>>   if N = 1 then
>>>     return d;
>>>   else
>>>     E = FFT(N/2, (a0, a2, ..., aN-2));
>>>     O = FFT(N/2, (a1, a3, ..., aN-1));
>>>     for k = 0 to k <= N/2 - 1 do
>>>       Ak = Ek + exp(-i2πk/N) * Ok;
>>>       Ak+N/2 = Ek - exp(-i2πk/N) * Ok;
>>>     return A;
```

Implementação

A natureza recursiva do algoritmo da transformada rápida de fourier o torna ideal para implementações em linguagens funcionais como Haskell. Abaixo exibimos as partes mais relevantes do código, onde omitimos inclusões de bibliotecas e a definição das funções auxiliares evens e odds.

```

--esse termo multiplica o somatório de termos ímpares
if xs n = exp $ -(0:+2*pi)*n / genericLength xs

--caso a função seja chamada com apenas dois termos, temos o caso base
ffti [x,y] n = x + y * (exp $ -(0:+pi)*n)

--caso contrário aplique a função recursivamente separando o somatório em
-- termos com índice par e com índice ímpar
ffti xs n = ffti (evens xs) n + f xs n * ffti (odds xs) n

--função que calcula os n coeficientes (ffti calcula um coeficiente)
ffft xs [] = []
ffft xs (y:ys) = (ffti xs y):(ffft xs ys)

--exemplo
ffft [0, 1, 4, 9] [0,1,2,3]

--saída ( :+ indica número complexo em Haskell)
[14.0 :+ 0.0, -4.0 :+ 8.0, -6.0 :+ 0.0, -4.0 :+ -8.0]

```

é embraiado

Embora não seja uma linguagem puramente funcional Wolfram Mathematica também se presta a uma implementação direta e clara.

```

// caso base de uma lista contendo apenas dois termos
ffft[{x_, y_}, n_] := x + y Exp[-I Pi n]
// caso contrário, divide a lista entre termos com índice ímpar e par
// e aplique a função recursivamente
ffft[f_List, n_] :=
  fft[Downsample[f, 2, 1], n] +
  Exp[-((I 2 Pi n)/Length[f])] fft[Downsample[f, 2, 2], n]
// acima calcula-se com um coeficiente abaixo calcula-se todos
ffft[f_List] := Table[ffft[f, n] // N, {n, 0, Length[f] - 1}]

// exemplo
ffft[{0, 1, 4, 9}]
{14., -4. + 8. I, -6., -4. - 8. I}

```

*ffft divide
usando expo
de arks?*

Exemplos

Abaixo encontra-se alguns exemplos básicos da transformada de fourier com base no código acima porém com modificações para lidar com o caso em que o intervalo de amostragem é diferente de $[0, 2\pi]$ e cuja taxa de amostragem é qualquer (diferente de $\frac{2\pi k}{N}$ como assumido acima)

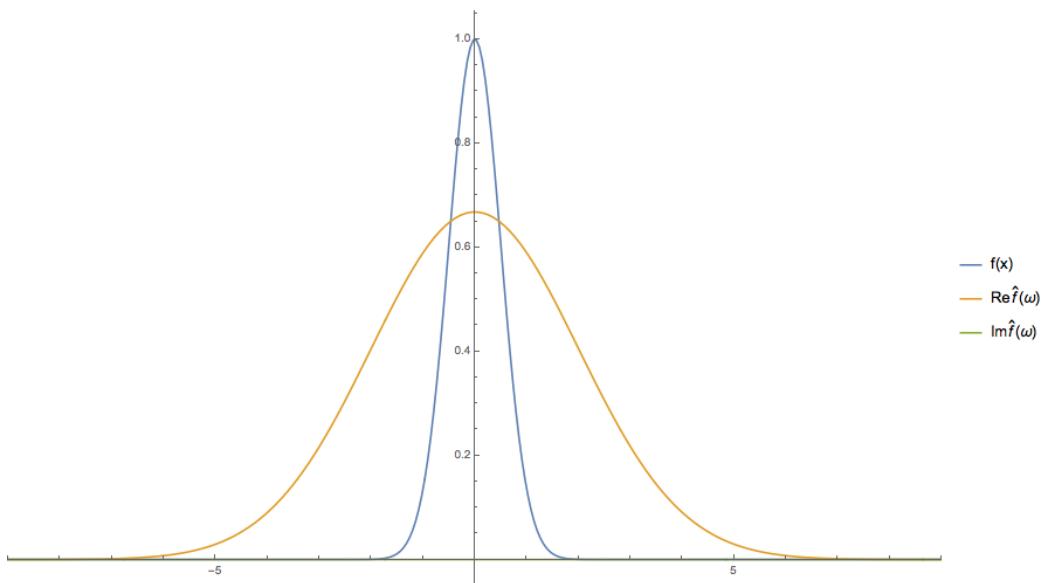
Gaussiana

A menos de uma normalização a função gaussiana é definida por: $f(x) = e^{-\frac{(x-\langle x \rangle)^2}{2\sigma^2}}$ Aplicando a transformada de fourier na curva gaussiana obtém-se outra gaussiana.

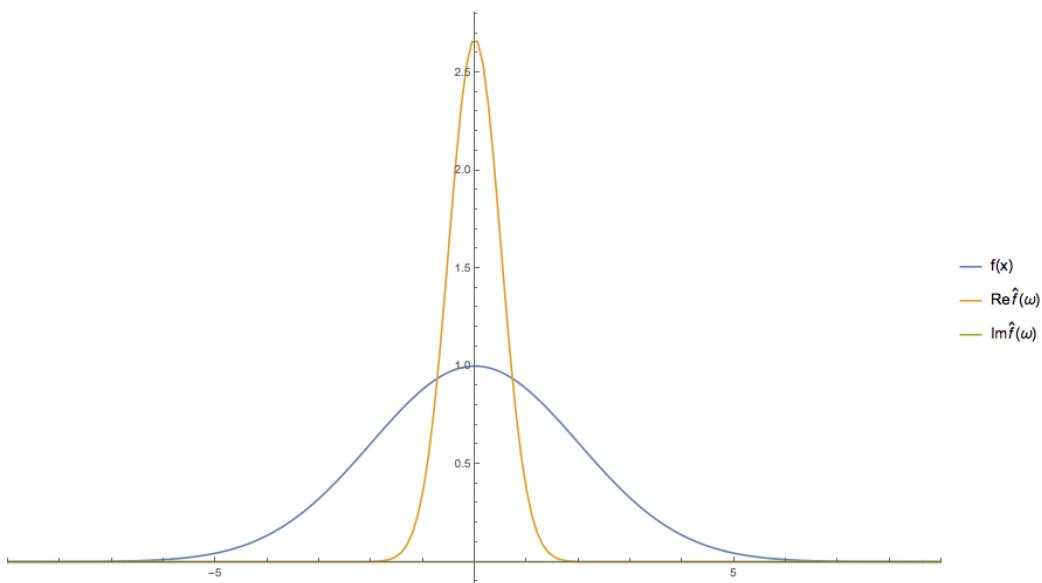
*qual das
óditas?
- com alguma
faz sentido?*

curva \rightarrow

f_k

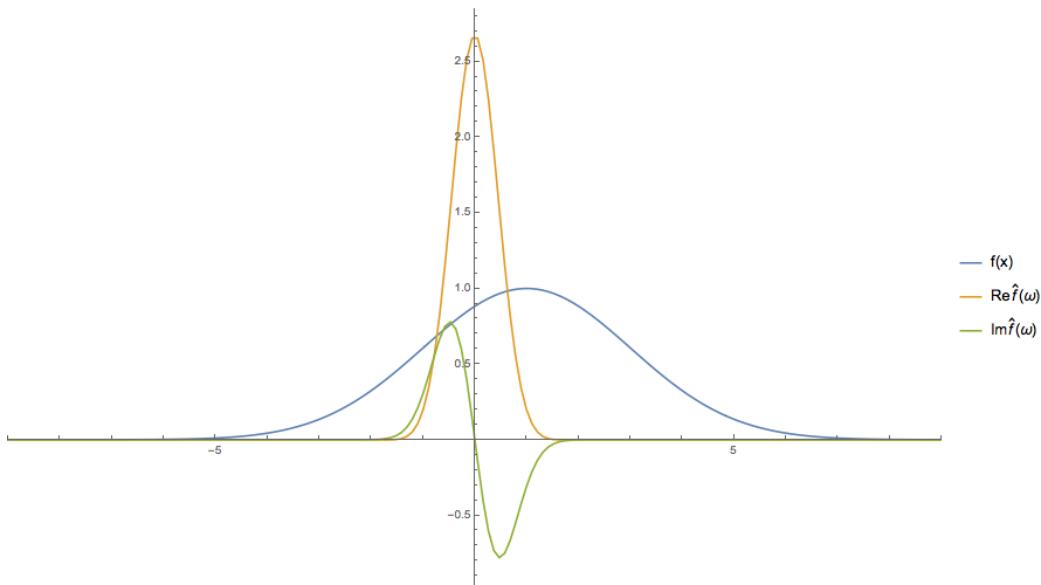


Observa-se que uma curva larga no espaço real corresponde a uma curva estreita no espaço de fourier e vice-versa. Esse é um fato matemático amplamente conhecido e se exprime também na física pelo Princípio da Incerteza de Heisenberg. Abaixo uma curva gaussiana mais larga que a anterior e sua transformada correspondentemente mais estreita.



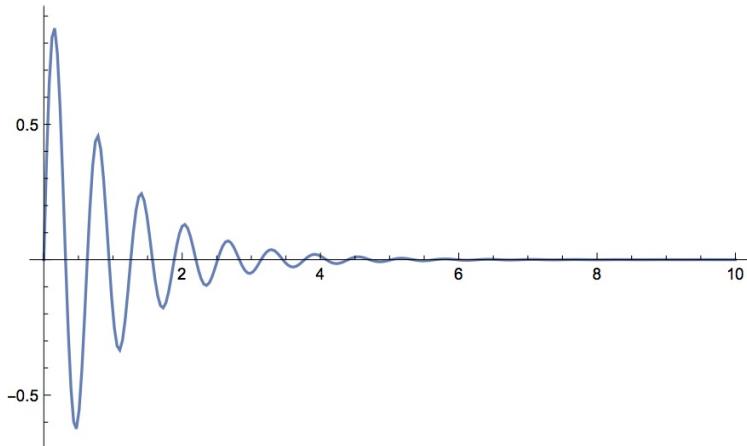
Uma função par em relação à origem possui uma expansão em fourier com termos ímpares nulos, no entanto, deslocando a gaussiana um pouco para a direita e, portanto, quebrando sua simetria, vemos que a sua transformada possui uma parte complexa:

Cela a inflexão e os resulhos extatos



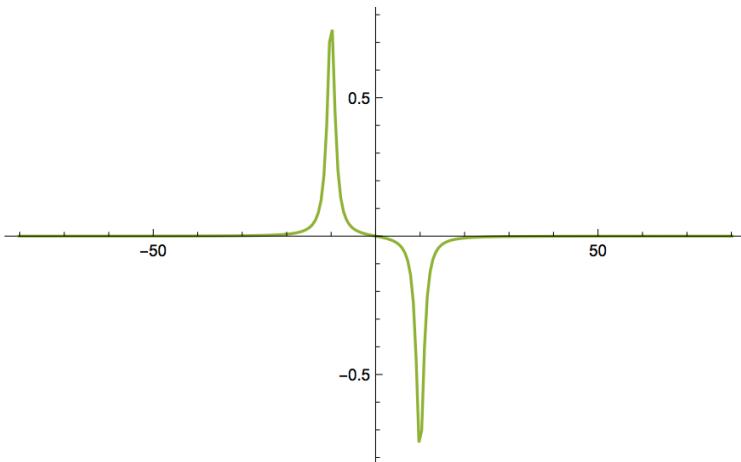
Oscilador Harmônico Amortecido

A curva que representa um oscilador harmônico amortecido é descrita por $f(x) = \sin(ax)e^{-bx}$ onde o termo senoidal, periódico, é amortecido pelo termo exponencial que leva a oscilação rapidamente a zero.



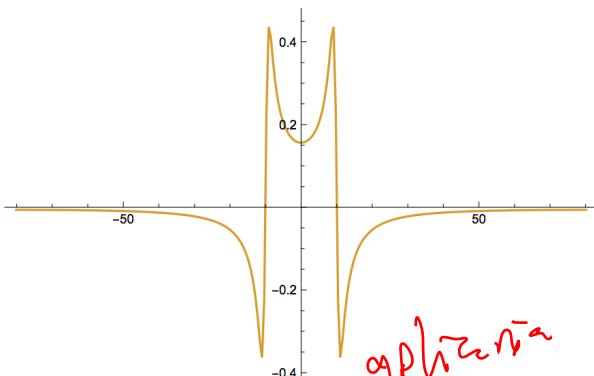
queis os
período
figura?

A transformada de fourier dessa curva possui parte real e imaginária, da parte imaginária dessa curva é possível obter a frequência de ressonância do fenômeno representado por ela. No caso abaixo vemos que a frequência de ressonância vale $\pm 10\text{Hz}$



interpretar essa análise dos resultados

Abaixo a parte real da transformada:



Aplicações

Equação de Schrödinger: propagação de onda de probabilidade

O cálculo da evolução temporal da equação de schrödinger é computacionalmente exigente. A cada passo de tempo é necessário recalcular a posição da onda de probabilidade que representa a partícula em cada ponto do domínio. Portanto, torna-se vantajoso o uso da transformada de fourier rápida. Além disso a própria transformada faz parte da solução, pois também deseja-se obter a sua evolução temporal no espaço de momento. O código foi desenvolvido em python por Jake Vanderplas (<https://jakevdp.github.io/blog/2012/09/05/quantum-python/>), no entanto, sob extensas modificações foi possível não apenas entender o código mas também modificá-lo para representar situações diferentes das propostas pelo autor. No site do autor encontra-se a explicação da teoria por trás do problema e também o funcionamento do algoritmo. A seguir exploraremos a interação de uma onda de probabilidade com potenciais de diferentes formas.

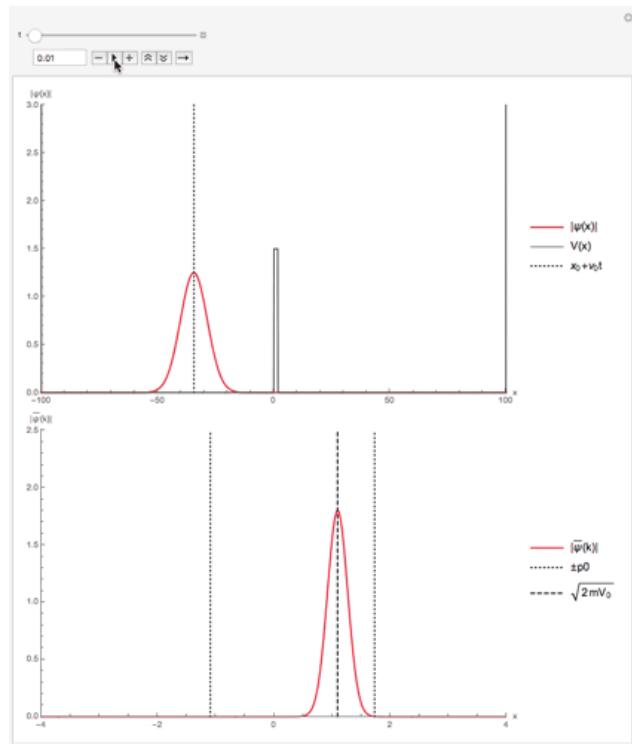
A parte principal do código modificado é exibida abaixo. Nota-se que o autor utilizou o método Leapfrog para integrar as equações de movimento fazendo a seguinte sequência de operações: dar um meio passo no espaço real, tomar a transformada, dar um passo no espaço de momento, tomar a transformada inversa e finalmente dar um passo completo no espaço real.

```
def time_step():
    global t, psi_mod_x, psi_mod_k, t_max
    psi_mod_x *= x_evolve_half
    for i in range(N_steps):
        psi_mod_k = fft(psi_mod_x)
        psi_mod_k *= k_evolve
        psi_mod_x = ifft(psi_mod_k)
        psi_mod_x *= x_evolve_half*x_evolve_half
    psi_mod_k = fft(psi_mod_x)
    t += dt * N_steps
    t_max = t_max - 1
```

O restante do código ocupa-se somente de inicialização de valores, definição do potencial e escrita dos resultados.

Potencial Quadrado

O potencial quadrado foi o exemplo escolhido pelo autor do código. Nesse exemplo fica visível que uma porção da onda de probabilidade atravessa o potencial embora classicamente não tenha energia suficiente para tal feito. Essa é uma demonstração numérica do efeito de tunelamento.



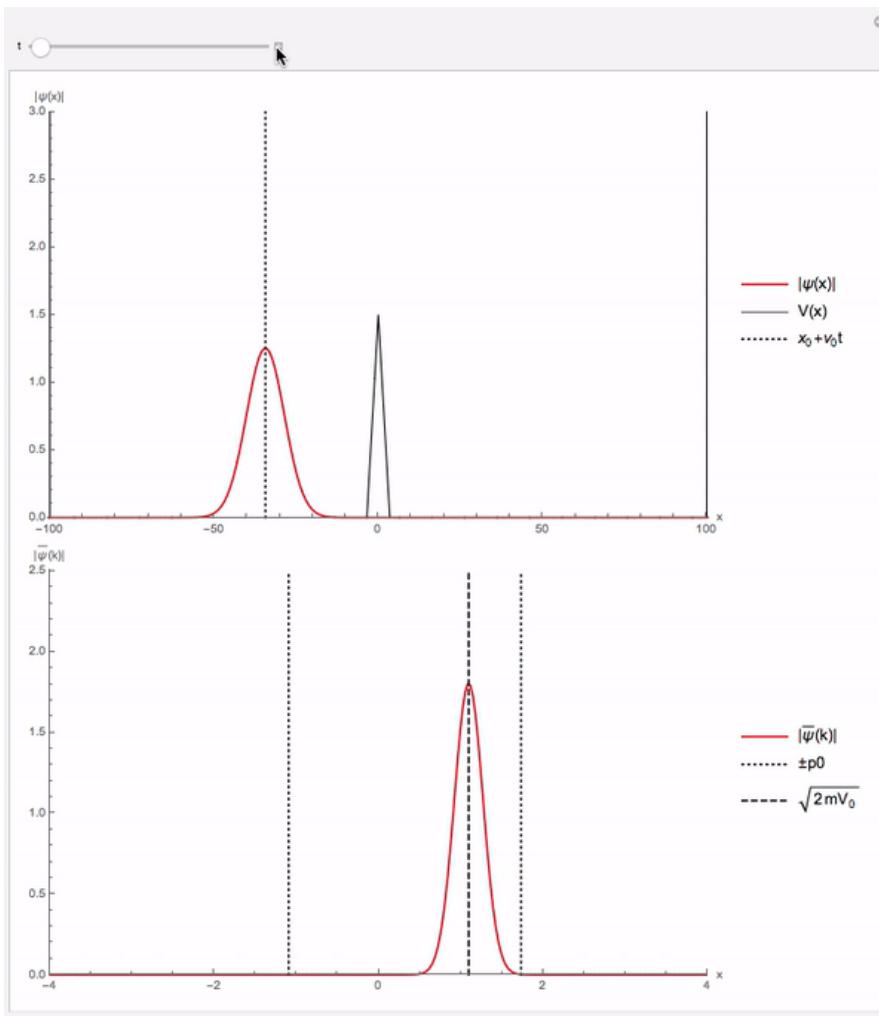
Coloca
as $\approx k$'s
comparação
flamejante.
for 1.
livre

Potencial Triangular

O potencial triangular, por ter uma base mais larga, coíbe o efeito de tunelamento, no entanto, uma pequena porção da onda incidente ainda consegue transpor a barreira.

o que é
o parâmetro
de incidente
da figura?

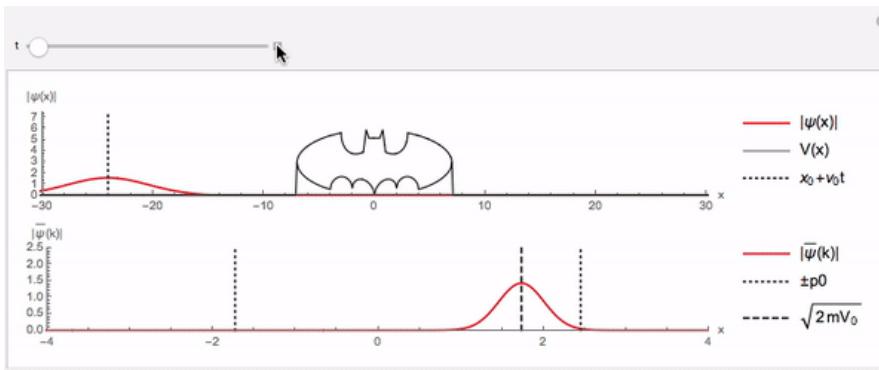
interpretando
comparação
verificado matemáticos



*l o po⁺
harmonico?
~ h ~ h
hidrogeno?*

Potencial Batman

Em cursos básicos de quântica normalmente resolve-se apenas o potencial quadrado, às vezes, excepcionalmente, apresenta-se o potencial triangular. Isso se deve a dificuldade em tratar casos de potenciais irregulares analiticamente. Mas em casos reais os potenciais são extremamente irregulares. A solução numérica trata tais casos com naturalidade e sem qualquer alteração. Em particular, um potencial tipo batman é encarado sem problemas:



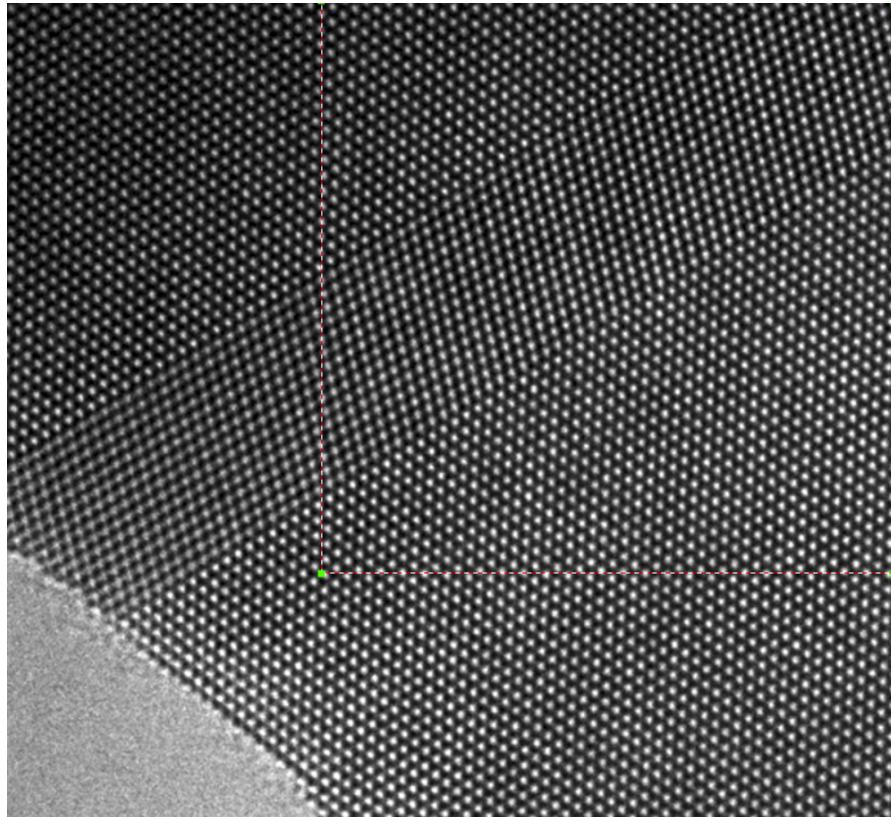
A parte superior da curva é apenas ilustrativa (pois não faz sentido matemático nem físico)

Padrões de Difração

A transformada rápida é muito utilizada no tratamento de imagens. Software como ImageJ ou Matlab possuem muitas ferramentas que fazem uso da técnica. Uma aplicação específica da FFT é na análise de imagens de partículas nanométricas obtidas por um microscópio eletrônico de transmissão. Esse equipamento produz um interferograma que (a menos de correções) representa a imagem real do que se está observando mas também produz uma segunda imagem, localizada no ponto focal da lente, que é o padrão de difração da imagem real. Essa imagem é uma visualização do espaço recíproco, ou seja, da transformada do espaço real.

(REF)

Abaixo está uma imagem de um plano de átomos de ouro obtida por um microscópio de transmissão eletrônica com uma escala de 2 nanômetros na qual destacamos uma região (quadrado vermelho) na qual tomaremos a transformada de fourier. A imagem em si é representada por uma matrix 800x800 onde cada elemento da matrix é um valor de intensidade sendo 0 para totalmente escuro e 255 para totalmente claro.



tem que colocar
de onde pegou
a figura...

pele mais
grossa

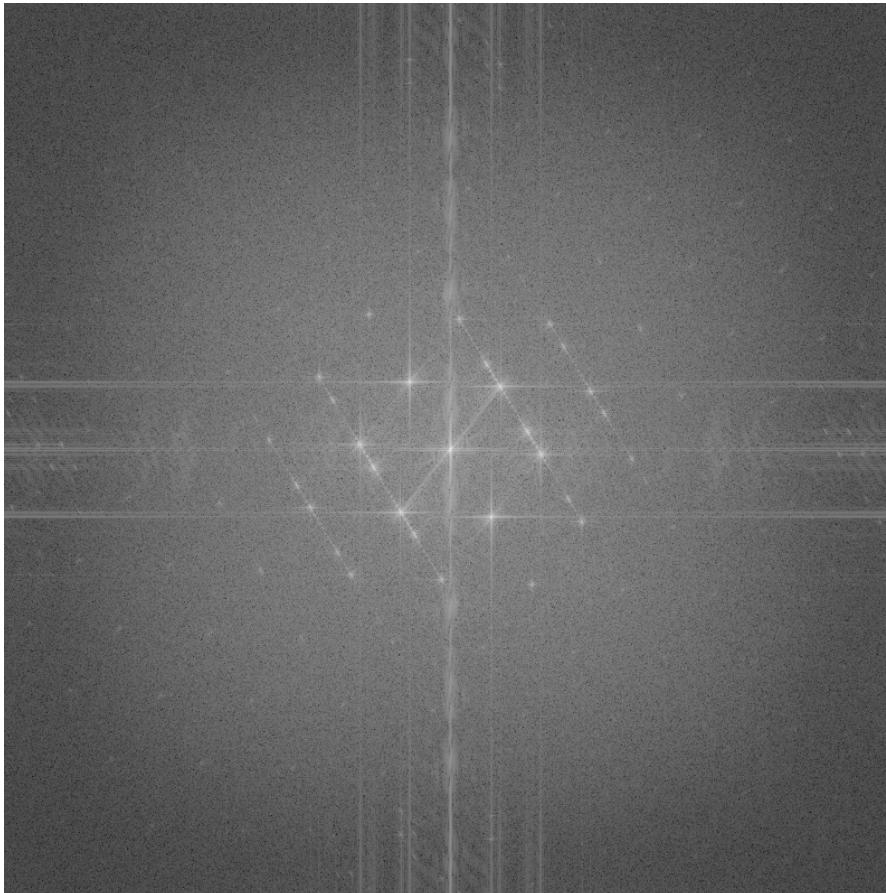
Utilizando, por exemplo o software Matlab podemos tomar a transformada inversa dessa imagem. A transformada de fourier produz o padrão de difração que revela a periodicidade da imagem. O código em Matlab é bem simples:

```
function imagefft(I)
F = fft2(I);
F = fftshift(F); % centraliza a FFT
F = abs(F); % obtém a magnitude (desnecessário, nossa imagem varia de 0 a 255)
F = log(F+1); % escala logarítmica, +1 por que log(0) é indefinido
F = mat2gray(F); % renormaliza a imagem de 0-255 para 0-1
imshow(F,[],) % exibe o resultado
end
```

• Não tem como colocar em python

ou scilab?

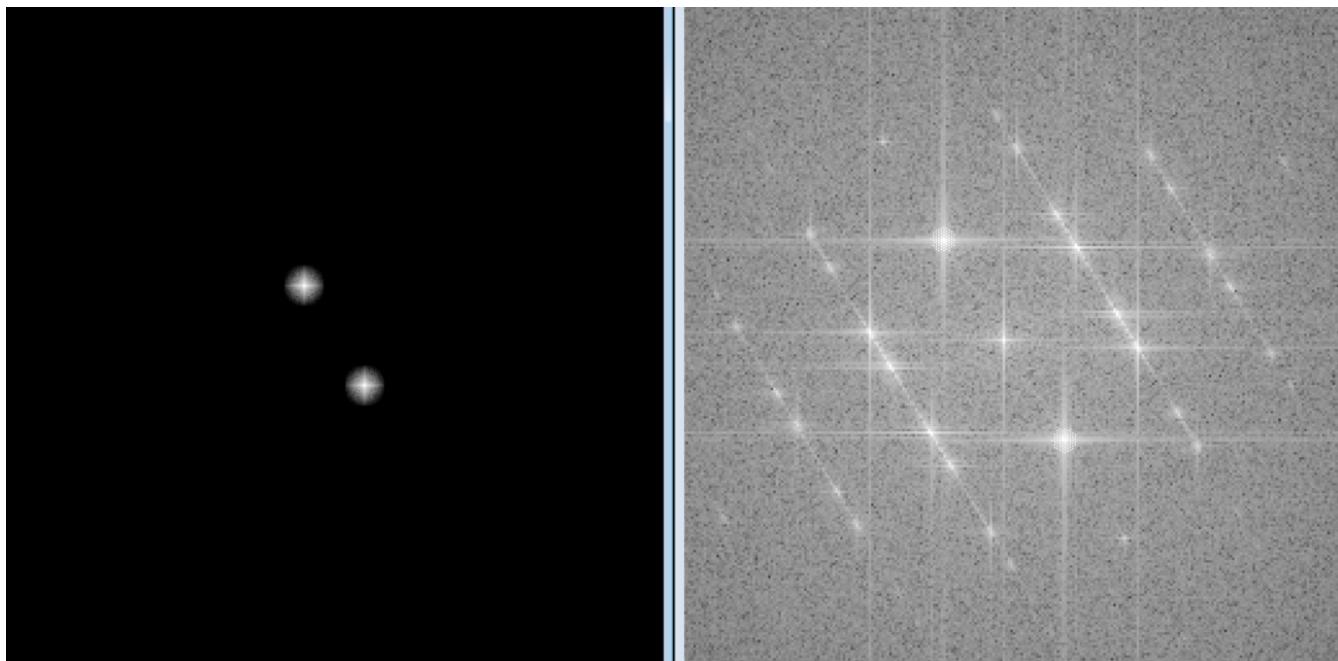
• Matlab é o software livre...



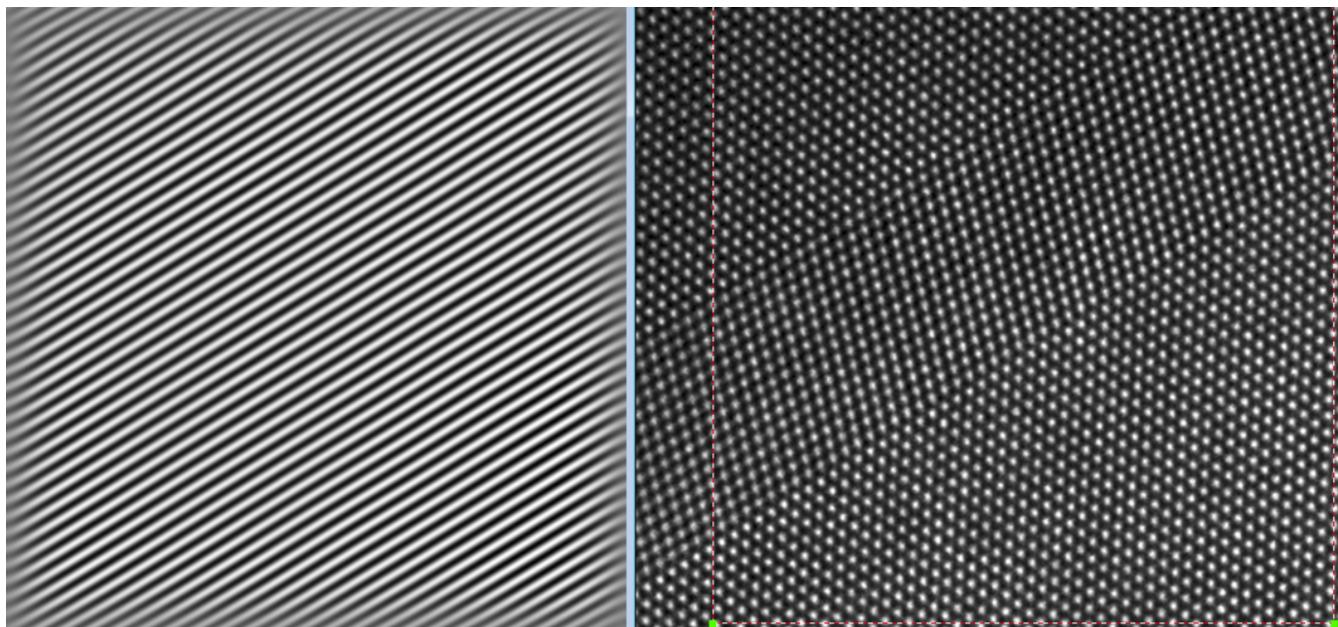
• indica os
planos
que figurem
e explique
melhor

Cada par de pontos simétricos em relação ao ponto central representa um conjunto de planos paralelos no espaço real, sendo a imagem real o conjunto de todos os planos representados por todos os pontos somado ao ruído da imagem. Observe que a imagem original, além de ruído, apresenta três planos diferentes como pode-se notar pela orientação dos átomos. Isso explica a presença de linhas claras ao redor dos pontos de difração. Um cristal perfeito, por outro lado, possuiria um padrão de difração de pontos localizados e não discos com linhas como vemos num cristal real.

A seguir vejamos como decompor essas informações. Aplicando uma máscara sobre a imagem podemos selecionar apenas um par de pontos e ver a qual conjunto de planos do espaço real ele está relacionado:

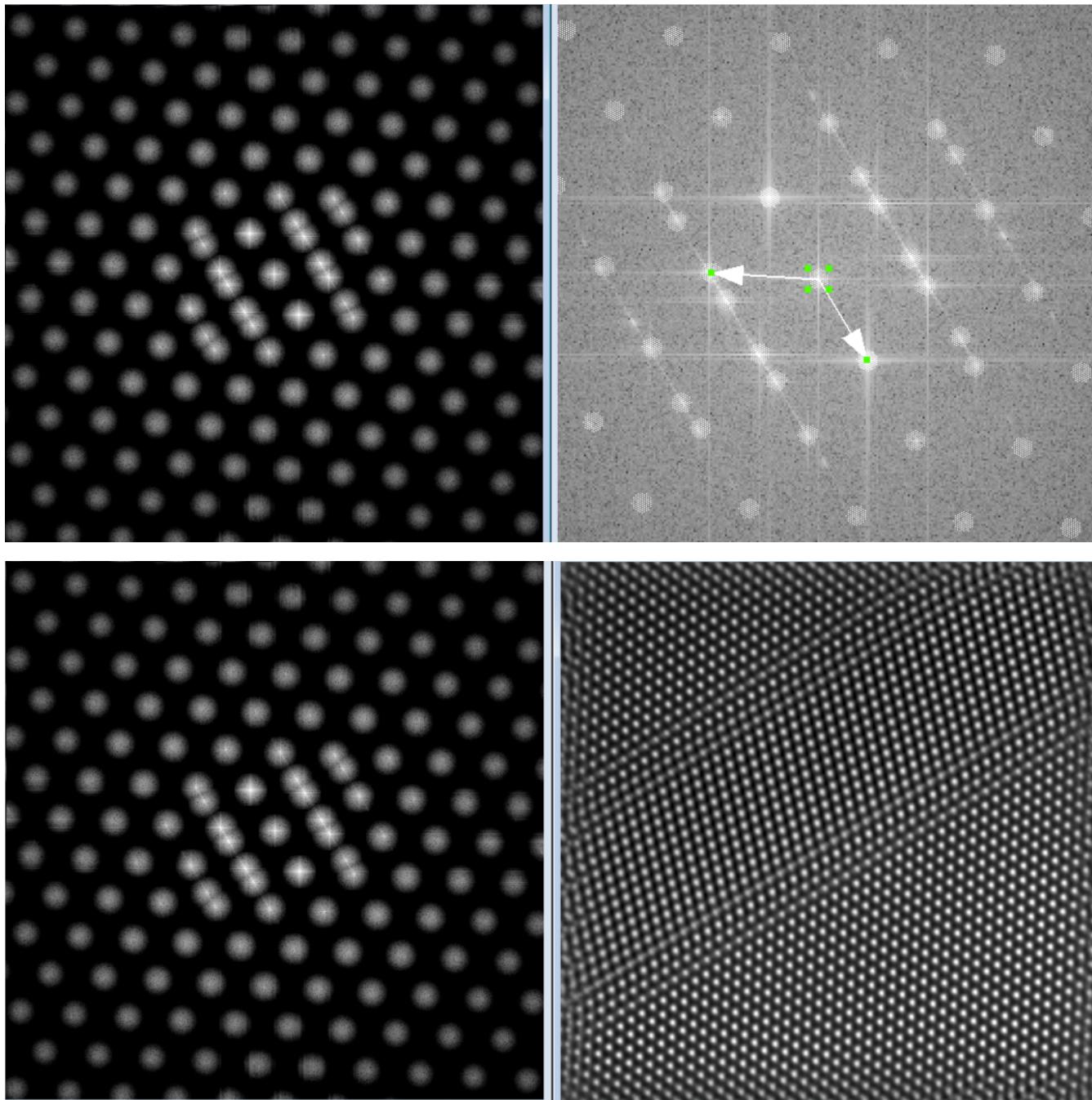


Para descobrir qual é o conjunto de planos tomamos a transformada inversa:



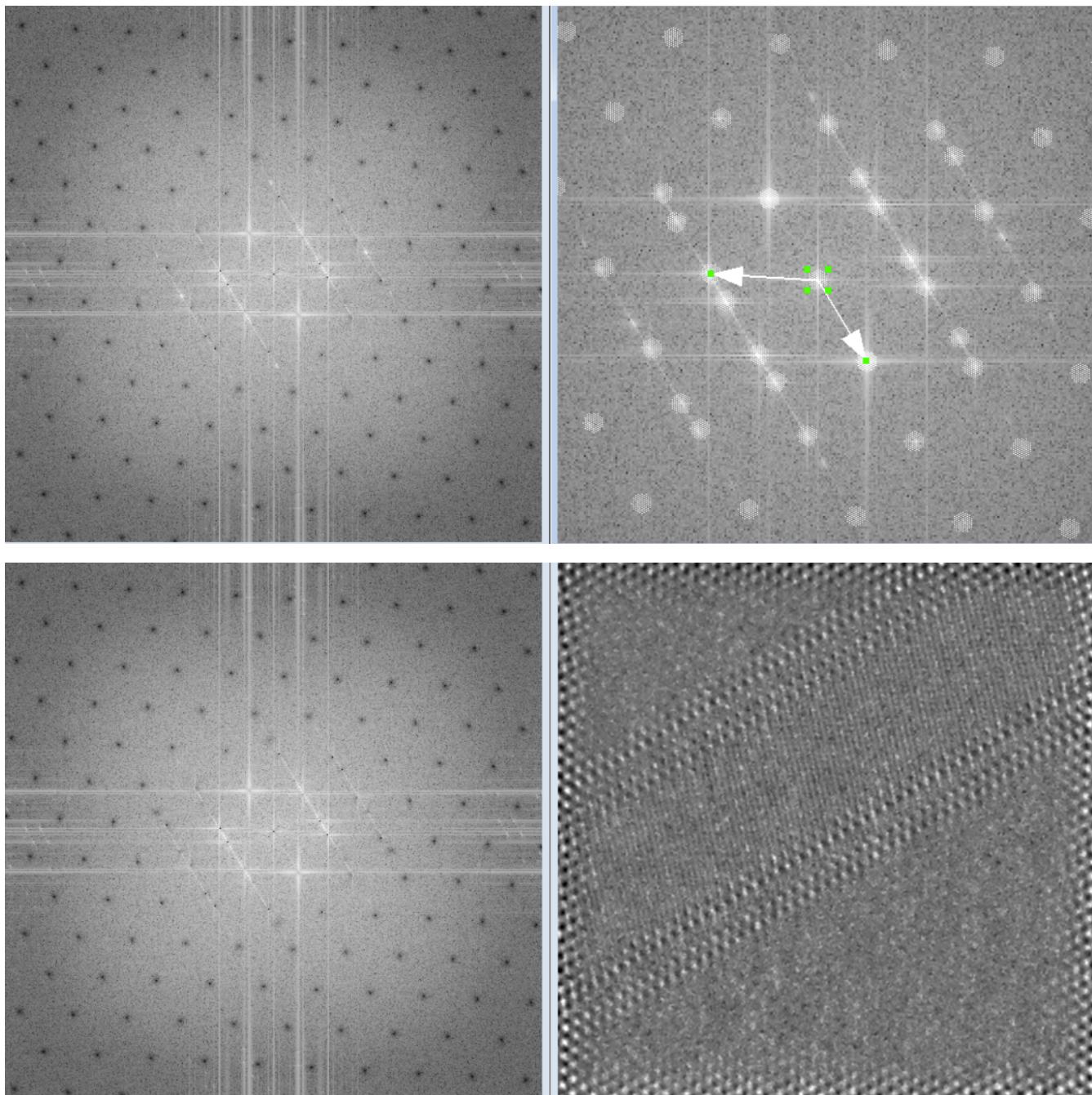
Como mencionado se selecionarmos todos os pontos e tomarmos a transformada inversa obteremos a imagem original, no entanto, removendo boa parte dos ruídos - a parte que deixamos de fora da máscara.

wish bon !



Se quisermos ver a composição do ruído podemos tomar a máscara inversa e aplicar a transformada inversa, obtendo:

• explora melhor
a aplicação da
máscara -
e o que é
comum



*explorar ~
poco
poco*

Referências Bibliográficas

- [1] An Algorithm for the Machine Calculation of Complex Fourier Series. By James W. Cooley and John W. Tukey (<https://web.stanford.edu/class/cme324/classics/cooley-tukey.pdf>).
- [2] Kreyszig, Erwin. Advanced Engineering Mathematics. Wiley, 2011.
- [3] Scherer, Claudio. Métodos Computacionais da Física.
- [4] Richard; Faires, Douglas. Numerical Analysis, 2011

Disponível em "http://fiscomp.if.ufrgs.br/index.php?title=Grupo4_-_FFT&oldid=1011"

-
- Esta página foi modificada pela última vez à(s) 21h10min de 29 de outubro de 2017.
 - Esta página foi acessada 200 vezes.

• No contexto de EPP
é melhor falar que a FT
“simplifica” o problema...